DECT Shield for Arduino

Marcelo de Jesus Pardal Vicente
Instituto Superior Técnico
INESC-ID / IST / UTL Lisboa
Email: marcelovicente@ist.utl.pt

***Abstract*—Lack of low-cost and accessible DECT development resources and materials makes it difficult to be used by independent developers and hobbyists when compared to other short-range wireless technologies (e.g. ZigBee, Bluetooth).**

This paper proposes an inexpensive Arduino shield equipped with DECT technology. Arduino wide availability, online resources and community support makes it an ideal choice to bring DECT to the general public.

The developed hardware consists of fully functional DECT shields, allowing audio and data communications, and reproduction of audio files from a microSD card. Distributed software comprises of an Arduino API to interact with shield components in a comprehensive way. Tests executed cover all functionalities and determine the shield performance. Crowded and non-crowded DECT environments are used to verify correct shield operation.

***Index Terms*—Arduino, DECT, SPI, Wireless communications**

I. Introduction

TODAY'S world thrives on wireless communications. During the past years, the number of wireless capable devices has been growing rapidly and the range of applications has increased. Nowadays, they are used in areas which were mainly dominated by wired solutions, for example in sensor networks for home automation such as alarm systems, remote doors and wireless sound systems.

DECT [1] is the second most successfully ETSI standard after GSM, dominating the wireless voice application market with a share of 73% [2]. It was created primarily for cordless phone systems in Small Office/Home Office (SOHO) applications, but currently it is also used in a wide variety of voice and data applications, such as baby monitors and traffic lights [3]. DECT standard ensures high quality voice transmission without interference up to 100 meters of distance (no obstacles), not using the highly congested 2.4 GHz spectrum, but the licensed 1.8 GHz band.

Although DECT products are widely commercialized, there are not many resources available for independent developers who wish to use it, when compared with other wireless technologies (e.g. GSM, Wi-Fi, Zigbee, Bluetooth). DECT development boards and kits are relatively expensive and hard to acquire, requiring direct contact with manufacturers.

The main motivation behind this project is the lack of low-cost DECT development resources and materials. The system implemented tries to solve this problem by bringing DECT to the general public using Arduino as intermediary.

Arduino [4] is a low-cost open-source hardware development platform being widely used among hackers, hobbyists and scientists for small and medium size projects [5]. The key aspects of the Arduino are its simplicity, ease of use, fast prototyping and expansion capabilities. High popularity in recent years resulted in a vast number of daughter boards (shields) that add new functionalities to the Arduino, including wireless support.

By combining the Arduino with a shield containing DECT technology, it is intended to create a cheap solution which is capable of providing DECT functionalities, such as data and audio transmission, in a familiar way. Hopefully, this system will aid developers in applications which were previously limited by other wireless technologies.

The final system should be able to serve as a fast prototyping method for application which requires audio support. Applications such as intercommunication devices or audio and data relays should be easily implemented without spending much time programming and reading documentation.

The structure of this paper is as follows. Section II describes and explains the DECT standard and the Arduino features. Section III describes the shield implementation and development, including the final solution. Section IV presents the test applications developed and the analysis or the shield performance. Finally, the paper is concluded in Section V.

II. Technologies and Applications

A. DECT

DECT [6] is a standard specified by the ETSI for wireless communications developed in Europe in 1992. DECT typically works in star topology formed of base stations (FP) and portable terminals (PP). In order for a cell to exist, it requires a base station and at least one portable part which the station will serve. DECT defines the radio interface between the portable parts and the fixed part.

DECT relies on a method called Dynamic Channel Selection (DCS) for the channel allocation procedure [7]. Instead of using a fixed channel for communication, the

portable parts are continuously sensing the available channels and trying to use the best one. This method relies on the portable parts to select the channel instead of the fixed part. The portable terminal chooses the least interfered channel from a periodically updated list of all 120 existing channels. DECT is also capable to handover time slots in order to improve communication quality in a seamless way. This channel allocation method permits DECT to adapt automatically to the environment conditions. DECT uses the licensed 1.88-1.90 GHz spectrum therefore only DECT applications are allowed to use it. Table 1 show the list of DECT characteristics.

TABLE I
DECT CHARACTERISTICS [6]

Frequency band	1880-1900MHz
Number of carriers	10
Carrier spacing	1.728 MHz
Access technique	TDMA, TDD, FDMA
Traffic duplex channels	12
Data rate per channel	32 kbps
Range	30 m or 300 m
Modulation	GMSK (BT=0.5)
Sensitivity	-86 dBm at 0.01 BER
Average RF power / slot	10 mW
Peak RF power	250 mW

DECT employs 10ms frames, each with 24 slots (TDMA). The slots are separated into two groups of 12 slots: one group for the uplink (PP to FP) and the other for downlink (FP to PP). A duplex DECT channel is a combination of one uplink and one downlink slot (TDD). DECT supports up to 120 duplex channels as result of 10 different frequencies (FDMA) and 12 duplex channels per frequency.

The slots contain 320 bits of user data which can be used to handle voice communication at 32 kbps in ADPCM format or to transmit data up to 24 kbps.

DECT advantages over other commercially available wireless technologies range from low power consumption, short communication distance (up to 300 m), reliable voice communication, but provides smaller data rates. DECT is the only standard which provides short range voice communication natively.

B. *Arduino*

Arduino [8] (Fig. 1) is an open-source electronics prototyping platform base on flexible, easy-to-use hardware and software. It is intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

There are twenty different official prototype boards, fully supported and documented by the Arduino development team. 8-bit Atmel megaAVR processors [9] are used to provide on-board input and output support. A boot loader is used to transfer new sketches, an Arduino program, through an USB or RS-232 interface.

The software consists of an Arduino IDE, which is a multiplatform program developed in Java, used to write, compile and deploy sketches to the Arduino. The IDE comes with several C/C++ libraries providing a base set of functions targeted for hardware sensing and controlling. Sketches are compiled using a custom GCC compiler.

Arduino boards follow the same layout and specifications regarding pin placement, allowing the connection of daughter boards, or shields, without incompatibility issues. The Arduino shields are stackable hardware boards that connect to the Arduino using the pin headers, providing new functionalities.

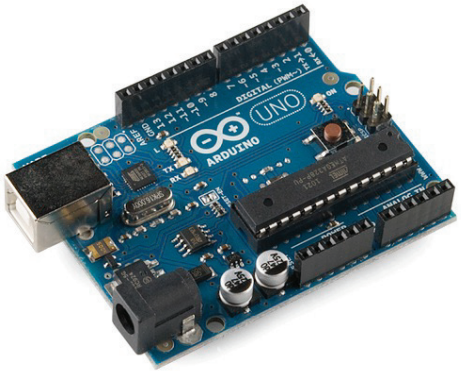

Fig. 1. Arduino Uno (rev. 3).

C. *Bithium DECT Module*

The Bithium DECT Module (BDM) is a small device which contains a microcontroller, a DECT radio transceiver and audio/digital outputs. C programs can be written in order to operate the module and it is capable of working as a fixed or portable part by changing the firmware. All DECT related procedures (e.g. registration, data and voice communication) are supported by the module.

The microcontroller in the BDM has a 16-bit architecture and operates at 82.944 MHz, being much faster than an Arduino.

The BDM uses an External Network Interface (ENI) layer, which is a proprietary software library developed by Bithium optimized for this device. The ENI contains a DECT stack with all necessary lower level layers (PHY, MAC, DLC, NWL and IWU) and defines a set of functions to interact with it.

III. IMPLEMENTATION

A. *Requirements*

The main goal of this project was to develop a shield that allows the Arduino to use the functionalities supported by the BDM. The shield had to be cheap, easy to use and support attractive functionalities for wireless applications. It was conceptualized after analyzing the DECT specifications, the BDM, the Arduino and the commercially available wireless shields.

A list of requirements used during the implementation is presented:

- Use of Bithium DECT Modules for DECT operations.
- Ability to use the shield as a DECT fixed or portable part.
- Ability to perform DECT calls and data transfers between shields.
- Ability to reproduce and transmit an audio stream.
- Support for a headset and a loudspeaker.
- Ability to regulate speaker volume and microphone gain.
- Bidirectional communication between the Arduino and the shield.
- Easy to use library for Arduino.
- Support for asynchronous events.
- Shield dimensions cannot be much larger than the Arduino Uno.

B. *Architecture*

The shield is composed of three main components (Fig. 2): a BDM, a microSD card slot and a custom SPI interface circuitry.

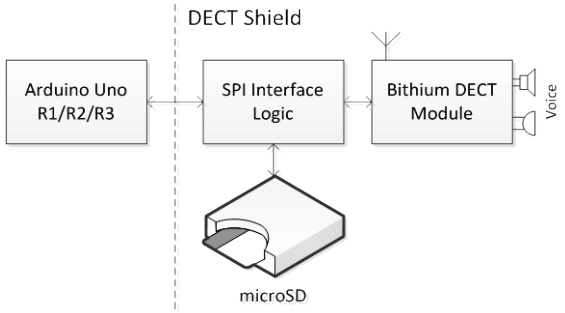

Fig. 2. DECT Shield architecture.

The microSD [10] slot complements the shield, allowing to store files in an external memory card. Files can be accessed by the Arduino or by the BDM. The BDM is able to open audio files from the card and play them without using the Arduino as an intermediary.

An SPI [11] circuitry interconnects the BDM and the microSD card with the Arduino. SPI allows fast data communication between devices but requires a higher signal counter compared to other protocols (e.g. UART, I2C [12]). The Arduino IDE already includes libraries to interface a microSD card using the Arduino SPI ports. Having the BDM connected using SPI resulted in a lower number of pins required, because it was already necessary for the microSD card.

The BDM is controlled using a custom message protocol with low memory footprint and with error checking mechanisms to enhance reliability.

C. *Hardware*

The shield hardware involves the connection of all components electrically. Shield components work at 3.3 V and an on-board linear regulator is used to convert 5 V provided by the Arduino. The BDM uses 1.8 V internally, but the I/O pins can work in pull-up mode, increasing the voltage range up to 3.5 V. All pins used in the BDM are configured to pull-up mode with the exception of the SPI pins to allow higher transfer rates. Using pull-ups result in slower transition times for digital signals. A logical transceiver is used to boost the SPI signals from 1.8 V to 3.3 V. Arduino detects logic signals up to 3 V, so direct interface with BDM 3.3 V outputs is possible.

Four tri-state buffers are used to interface the SPI bus between the Arduino and the shield (Fig. 3). The buffers allow placing the shield SPI outputs and inputs in tri-state when the BDM is accessing the microSD. The buffers are tolerable to 5 V and the Arduino SPI output ports could be connected without needing additional components.

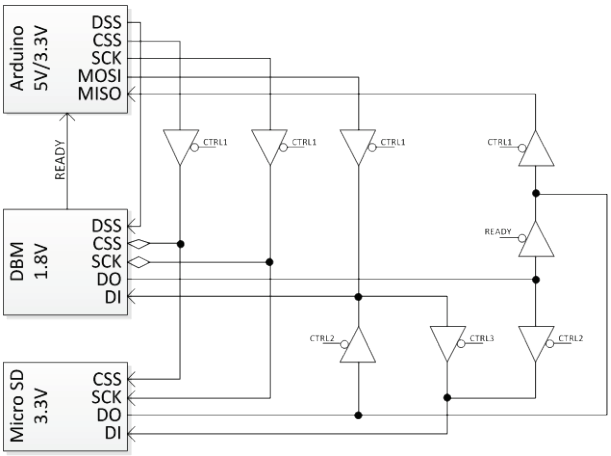

Fig. 3. SPI tri-state buffer schematic.

The BDM needs to periodically commutate from slave (Arduino commanding the shield) to master operation (BDM accessing the microSD). Four additional tri-state buffers had to be added in order to allow this commutation, since the data-in (DI) and data-out (DO) pins in the BDM do not change direction like the ones in the Arduino do. The buffers prevent short-circuits and allow to commute the SPI data lines, establishing a channel.

These SPI buffers are controlled by the BDM, ensuring less signals come from the Arduino and more shield independency. The shield is in tri-state by default and the Arduino can request shield access by asserting the BDM selection signal (DSS). An additional synchronization signal (READY) is used to indicate when the BDM is ready to receive data from the Arduino and that the SPI channel is formed.

The maximum SPI clock frequencies, for each possible channel, were calculated considering the delays imposed by the devices (setup and hold times), buffers and logic transceivers (transition times). Clock frequencies can go up to 16.6 MHz (Arduino to BDM), 15.7 MHz (Arduino to microSD) and 14.7 MHz (BDM to microSD).

The shield was design to be easily connected to a headset. Audio connections consist of two standard 3.5 mm jacks for a mono speaker and a microphone, and a two-way terminal for a low impedance loudspeaker. Each audio connection contains

an on-board filter to improve audio quality. The audio out jack supports speakers with at least 28 Ω and at least 4 Ω for the loudspeaker connection.

An ISP is present in the shield and allows the programming and debugging of the BDM using UART at 1.8 V. There is also a status LED which indicates if the shield is working properly.

The shield layout (Fig. 4) was designed using two-layers for lower manufacturing costs and to be a possible DIY project. The DECT antenna was placed farther way from pin headers or other metallic components to mitigate as much interference as possible. The SPI logic circuitry was placed between the Arduino SPI pins, the microSD slot and the BDM SPI pins, ensuring that there was enough space for routing. The analog signals for audio were kept isolated as much as possible from digital signals. Finally, trace-free areas were left with cooper connected to Ground, improving power distribution.

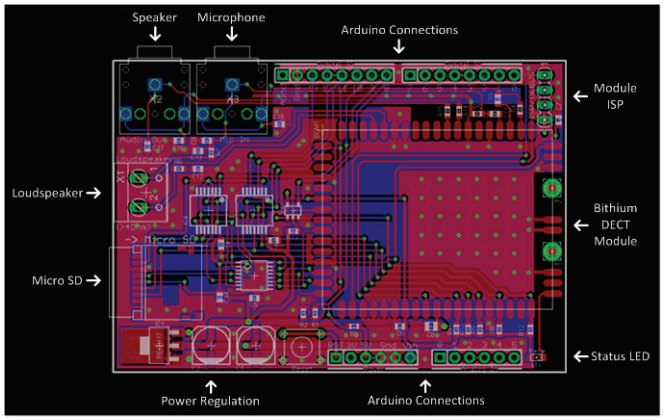

Fig. 4. DECT Shield layout (rev. A).

During the assembly of the PCB (Fig. 5) some minor layout errors were detected and corrected on a posterior revision.

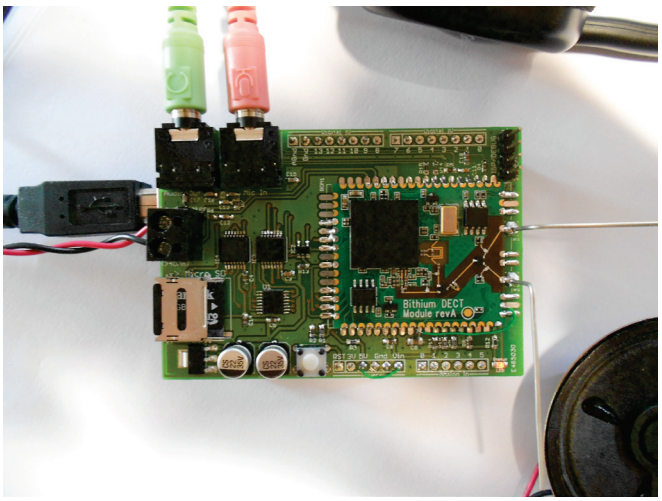

Fig. 5. DECT Shield assembled.

The shield uses Arduino pins 8 through 13 and can optionally use pin 2 to generate interrupts for certain events. The remaining pins can be used with other shield.

D. *Software*

The implemented software components consist of an Arduino DECT API to interact with the shield and two custom firmwares (FP and PP) for the BDM. Fig. 6 illustrates an overview of the communication path between the Arduino and the DECT Shield.

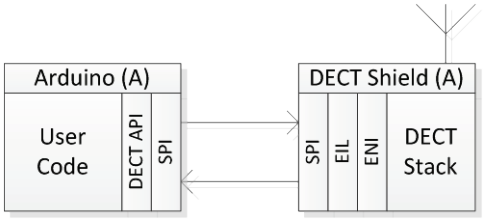

Fig. 6. Overview of Arduino and DECT Shield communication.

The Arduino API and the BDM firmwares have access to a set of functions and commands from a library named External Interface Layer (EIL), which defines the communication protocol between the two devices. The Arduino can control the BDM using any command defined by the EIL. The command set is separated in several groups, covering specific functionalities: Module, DECT, playback, audio and microSD.

Module commands control general BDM behavior (e.g. firmware, reset, configurations). DECT commands are used to perform DECT related actions (e.g. registration, data transfer, calls). Playback commands are used to control file reproduction from the microSD. Audio commands control the external audio connections (e.g. enable / disable speaker / microphone, volume, microphone gain). Finally, microSD commands control the memory card access.

The EIL present in the BDM interacts with the ENI by converting all DECT commands received from the Arduino to valid DECT operations. The remaining commands do not interact with the DECT stack or the ENI.

Communication is performed through SPI using a custom protocol system (Fig. 7), which encapsulates commands. Each message is separated in three parts: header, body and termination. The header indicates when a new message is starting, the command being sent and the size of the payload. The body contains the payload data associated to the command in the message. The body can carry up to 255 bytes of useful information. The termination contains a checksum to verify if the message was received without errors, and a termination indicator. The BDMs send a response back to the Arduino, using the same message system, after executing the command.

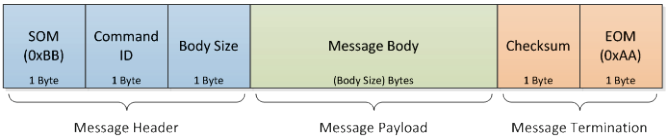

Fig. 7. SPI protocol message.

The Arduino DECT API library encapsulates the EIL and is used in Arduino sketches to interface the DECT Shield. Every

function present in the DECT API uses one or more commands from the EIL to control the shield. The API was written in C++ and supports pooling and events programming methods. Pooling requires the Arduino to be periodically querying the BDM for status updates, while events allow the Arduino to perform other tasks and when something important happens (e.g. new call, new data packet), it automatically executes a callback function associated to the event.

A microSD driver [13] and a FAT32 file system [14] emulator were developed in order to allow the BDM to reproduce audio files. The BDM accesses directly to the microSD, without using the Arduino as an intermediary, to read audio files in WAV format [15]. Files are interpreted by the BDM and raw audio data is outputted in the speaker output.

A mechanism is in place to prevent audio reproduction from stalling when the Arduino wants to send messages through the shared SPI bus. In order to provide a large time window for the Arduino to complete a transfer, the BDM only allows the Arduino to send a message after a data block is read from the microSD card.

A SPI Flow Control layer in the BDM controls and monitors the flow of the SPI data through the custom SPI circuitry. This layer changes the BDM SPI peripheral to master operation, when accessing the microSD, and to slave, when receiving commands from the Arduino. It is also responsible for controlling the buffers and transceivers assembled in the shield.

Fig. 8 illustrates an overview of all software abstraction layers present in the Arduino and the BDM.

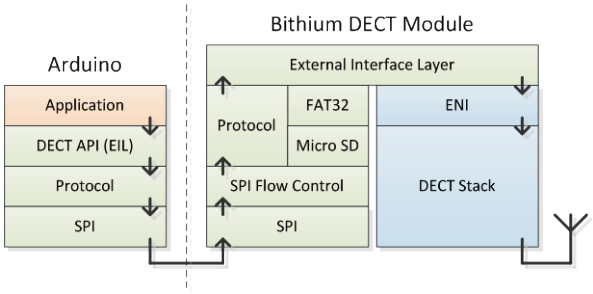

Fig. 8. Software abstraction layers.

IV. Test Applications

A total of three test applications were written for the Arduino. All functionalities available by the shield are covered by the combination of all three applications.

A. Terminal

All functionalities with the exception of the events were tested using a bash-like application for the Arduino. The Terminal allows the user to input commands using the serial port, which in turn invokes functions from the DECT API. All functions present in the API are covered by this sketch. Using this application was possible to test the registration procedure, data transfer, voice communication, external connections and the playback of audio files.

The code of the Terminal sketch exemplifies how to use all functions in the API in a generic application.

B. Text-to-Speech

The Text-to-Speech sketch was used to test audio reproduction in the BDM and the simultaneous transmission of commands from the Arduino and blocks from the microSD card.

The user can write words using the serial port and every time the sketch detects a space, it converts the last typed word into a file path (e.g. ‘h/e/llo.wav’ for the word ‘hello’). Before sending the path to the shield, the sketch pools the shield playback status periodically until there is no file playing. When the shield is free to play another file, the sketch relays the path to the shield and issues a play command.

A single microSD card was filled with 109 577 individual WAV files containing spoken words generated using AT&T Natural Voices [16].

C. Telephone

In order to test data and voice communication between shields, a rudimentary telephone sketch was developed, allowing to do internal voice calls.

The test application uses only events and simulates most features present in a normal telephone: ringing, waiting for answer, hang-up and a contact list.

Fig. 9 illustrates the state diagram of the sketch. The sketch is initialized idle and it is allowed to receive incoming calls. If an incoming call is detected, the sketches simulates a ringing, querying the user to accept or refuse the call. By accepting the call, an audio channel is formed between the two devices and voice communication becomes possible. Terminating the call returns the sketch to idle.

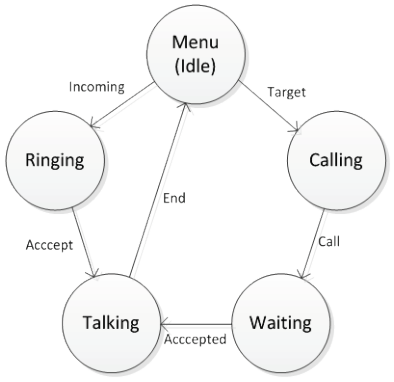

Fig. 9. Telephone sketch state diagram.

Data communication is used to request a name identifier from all registered DECT devices running Telephone, forming a contact list of possible targets to call (Calling state). A call is started by selecting one of possible targets from the list, which places the application in waiting state and the target ringing.

Using this test application it was possible to verify if the DECT components of the shields were working properly.

D. Shield Performance

A shield range test was executed in a crowed DECT environment. The test consisted in establishing several voice calls, using the Telephone sketch, until the shield stopped detecting the destination part. Tests showed the shields are able to perform voice calls without interference at least 30 meters, with no obstacles, and up to 10 meters, with obstacles.

Transmission rates were tested using sequences of data packets between two shields. The maximum transmission rate measured was of approximately 1 kbps. This value was expected because dedicated data channels are not supported by the BDM firmware and data communication is achieved using DECT control messages.

SPI communication between the Arduino and the BDM was executed at 1 MHz and at 10 MHz between the microSD card and the BDM. Commands sent from the Arduino were received correctly during file reproduction and no audio gaps were detected at these frequencies.

Tests show the shield consumes an average of 55.5 mA as portable part and 64.0 mA as fixed part, when using all features at the same time.

The DECT API library for the Arduino requires approximately 4 kB of programming space and 12 bytes of RAM. All functions and events were tested and work according to plan, but the library size is bigger than anticipated.

V. Conclusion

The main objective of this thesis was to specify, design, manufacture and assemble a DECT shield using a Bithium DECT Module with a comprehensible and easy-to-use library for Arduino in order to bring DECT to the general public. These objectives were achieved and the shield correct functioning was tested and demonstrated. A set of Arduino example sketches were developed to aid programmers understand the API.

By writing a sketch for the Arduino it is possible to command the BDM to register, call and transfer data to other compatible DECT devices. The shields are double layered printed circuit boards and their components are very common and easy to obtain, making it an undemanding DIY project for hobbyists and independent developers. The shield hardware and Arduino software are fully open-source and available for download. The BDM software code contains proprietary code and was not distributed.

All functionalities supported by the software were tested in crowded and non-crowded DECT environments using the three test applications developed for this purpose. All features present in the shields are covered by these applications and work as expected, allowing to reproduce files, register DECT devices and to transfer data and audio wirelessly.

The DECT Shield costs approximately 30 € and is within expected values, considering the functionalities implemented and the current shields in the market.

Not all functionalities were implemented due to resources and time limitations. Missing features and improvements should be easy to implement due to the modular programming model used in the BDM software and Arduino DECT API.

The shield hardware is working as intended but component placement and routing can be improved in future revisions. DECT functionalities such as call handovers, walky-talky mode and dedicated data channels are not yet implemented.

Playback is only being supported by a shield working as a fixed part due to firmware limitations. A new code needs to be written in order to grant support for portable parts. Support for additional audio formats can also be implemented.

Improvements to the Arduino DECT API are recommended. The library is occupying more programming space than anticipated and different programming models can be implemented and tested to obtain better results.

Finally, only two devices were used to test the software components. Additional testing using more than two devices is required to verify if the shields handle more devices correctly.

References

[1] *ETSI EN300 175: Radio Equipment and Systems (RES): Digital Enhanced Cordless Telecommunications (DECT)*, Parts 1-5, ETSI, 1992.

[2] University of Manchester, (2005), *DECT,* Available: http://www.cs.manchester.ac.uk/ugt/COMP38512/cs3091.

[3] A. Schuler, E. Tews, R.P. Weinmann, *deDECTed.org*, 2008.

[4] Arduino (2012), *Arduino Official Site,* Available: http://arduino.cc/.

[5] Hack n Mode, *Top 40 Arduino Projects of the Web,* Available: http://hacknmod.com/hack/top-40-arduino-projects-of-the-web.

[6] A. Muchaxo, A. Sousa, N. Pereira and H. Sarmento, *Wireless Data Communications Using DECT Air Interface*, ACM SIGCOMM Computer Communication Review, vol .29, April 1999.

[7] H. Eriksson, *Performance of dynamic channel allocation in the DECT system*, *Vehicular Technology Conference, 1991. Gateway to the Future Technology in Motion, 41st IEEE,* pp. 693 - 698, May 1991.

[8] Arduino (2012), *Arduino Uno,* Available: http://arduino.cc/en/Main/ArduinoBoardUno.

[9] Atmel (2012), *megaAVR,* Available: http://www.atmel.com/products/microcontrollers/avr/megaavr.aspx.

[10] Transcend, *microSDHC Card series*, 2009.

[11] Microchip, *SPI, Overview and Use of the PICmicro Serial Peripheral Interface*, 2002.

[12] Freescale, *I2C-bus specification and user manual*, 2012.

[13] SanDisk, *SanDisk Secure Digital Card*, 2003.

[14] Microsoft, *Microsoft Extensible Firmware Initiative, FAT32 File System Specification*, 2000.

[15] Microsoft, *Multimedia Programming Interface and Data Specifications*, 1991.

[16] AT&T, *AT&T Natural Voices® Text-to-Speech Demo,* Available: http://www2.research.att.com/~ttsweb/tts/demo.php.